We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 3

Simulated Quenching Algorithm for Frequency Planning in Cellular Systems

Luis M. San-José-Revuelta

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50566

1. Introduction

This chapter focuses on a specific application of two Natural Computation (NC)-based techniques: Simulated Quenching (SQ) and Genetic Algorithms (GA): the problem of channels' assignment to radio base stations in a spectrum efficient way. This task is known to be an NP-complete optimization problem and has been extensively studied in the last two decades. We have decided to include both SQ and GA in the chapter so as o give an academic orientation to this work for readers interested in practical comparisons of NC methods.

According to this point of view, our main interest is in describing and comparing the performances of these algorithms for solving the channel allocation problem (CAP). Therefore, the aim of our work is not the full theoretical description of SQ and GA. Surely, some other chapters in this book will cover this issue.

There exists an important research activity in the field of mobile communications in order to develop sophisticated systems with increased network capacity and performance. A particular problem in this context is the assignment of available channels (or frequencies) to based stations in a way that quality of service is guaranteed. Like most of the problems that appear in complex modern systems, this one is characterized by a search space whose complexity increases exponentially with the size of the input, being, therefore, intractable for solutions using analytical or simple deterministic approaches (Krishnamachari, 1998; Lee, 2005). An important group of these problems –including the one we are interested in– belong to the class of NP-complete problems (Garey, 1979).

Simulated Quenching (SQ) belongs to the family of Simulated Annealing (SA)-like algorithms. Simulated Annealing is a general method for solving these kind of combinatorial optimization problems. It was originally proposed by (Kirkpatrick, 1983) and (Černy, 1985). Since then, it has been applied in many engineering areas. The basic SA algorithm can be considered a generalization of the local search scheme, where in each step

of the iterative process a neighbour s' of each current solution s is proposed at random. Then s is only replaced by s' if cost does not rise. The main problems of SA are: (i) the possibility to get trapped in local minima, and (ii) the large computational load that leads to slow algorithms.

Both of the algorithms described in this chapter try to solve these drawbacks. The first one, SQ, speeds up the algorithm by quickly reducing the temperature in the system, though, in the process, the advantage of SA, i.e. convergence to the global optimum, is defeated (Ingber, 1993). Due to this, SQ is termed as a greedier algorithm in terms of computational load compromising for the global optimum. Besides, the proposed method, occasionally allows moves to solutions of higher cost according to the so-called Metropolis criterion (Metropolis, 1953) (see section 4.1). However, Duque et al. (Duque, 1993) point out that the main drawback they found was to get trapped to a local minimum (by a misplaced transition) with a low chance to get out of it.

The problem of convergence to suboptimal solutions can be efficiently addressed with another NC-based technique, the GAs. These algorithms are based on he principle of natural selection and survival of the fittest, thus constituting an alternative method for finding solutions to highly-nonlinear problems with multimodal solutions' spaces. GAs efficiently combine explorative and exploitative search to avoid convergence to suboptimal solutions. Unlike many other approaches, GAs are much less susceptible to local optima, since they provide the ability to selectively accept successive potential solutions even if they have a higher cost than the current solution (Mitchell, 1996).

This chapter focuses on the application of these methods to solve the channel allocation problem found in cellular radio systems. In this kind of systems, the frequency reuse by which the same channels are reused in different cells becomes crucial (Gibson, 1996). Every cell is allocated a set of channels according to its expected traffic demand. The entire available spectrum is allocated to a cluster of cells arranged in shapes that allow for uniform reuse patterns. Channels must be located satisfying certain frequency separation constraints to avoid channel interference using as small a bandwidth as possible. Considering this framework, the CAP fits into the category of multimodal and NP-complete problems (Krishnamachari, 1998; Garey, 1979; Hale, 1980; Katzela, 1996).

The *fixed* CAP –see section 3.2– has been extensively studied during the past decades. A comprehensive summary of the work done before 1980 can be found in (Hale, 1980). When only the co-channel constraint is considered, the CAP is equivalent to an NP-complete graph coloring problem (Sivarajan, 1989). In this simpler case, various graph-theoretic approaches have been proposed (Hale, 1980; Sivarajan, 1989; Box 1978; Kim, 1994).

From the point of view of NC, some procedures based on NNs (Funabiki, 1992; Hopfield, 1985; Kunz, 1991; Lochite, 1993) and SA or SQ (Duque, 1993; Kirkpatrick, 1983; Mathar, 1993) have already been considered. SA-SQ techniques generally improve the problem of convergence to local optima found with NNs, though their rate of convergence is rather slow (specially in SA), and a carefully designed cooling schedule is required. On the other

hand, several GA-based approaches have been applied to solve the CAP: For instance, (Cuppini, 1994) defines and uses specific genetic operators: an asexual crossover and a special mutation. A disadvantage of such crossover is that it can easily destroy the structure of the current solution and, thus, make the algorithm harder to converge. In (Lai, 1996), Lai and Coghill represented the channel assignment solution as a string of channel numbers grouped in such a way that the traffic requirement is satisfied. The evolution is then proceeded via a *partially matched crossover operator* (PMX) –this type of crossover has also been used in (Ghosh, 2003)– and basic mutation. Two years later, Ngo and Li (Ngo, 1998) suggested a GA that used the so-called *minimum separation encoding scheme*, where the number of 1's in each row of the binary assignment matrix corresponds to the number of channels allocated to the corresponding cell. Authors stated that this algorithm outperforms the NN-based approach described in (Funabiki, 1992).

The particularities of the GA shown in section 5 are, mainly, a low computational load and the capability of achieving good quality solutions (optimal, minimum-span, solutions) while maintaining satisfactory convergence properties. The probabilities of mutation and crossover of the GA are on-line adjusted by making use of an individuals' fitness dispersion measure based on the Shannon entropy (San José, 2007). This way, the diversity of the population is monitored and the thus obtained method offers the flexibility and robustness peculiar to GAs.

Another closely related problem that we are not going to deal with is the determination of a lower bound for the *span* (difference between the largest channel used and the smallest channel used) in channel assignment problems. For instance, (Mandal, 2004) and (Smith, 2000) address this problem when proposing solutions to the CAP.

The chapter is organized in the following sections:

- The first section (section 2) presents a brief introduction to the frequency (or channel) allocation problem. As mentioned above, there exist many good articles covering this problem, and our main purpose is to introduce the basic references of SQ and GAs applied to the CAP, to pose the problem as well as to describe a first motivation to research into this field.
- Next, section 3 explains mathematically the interference and traffic constraints that define the CAP.
- Sections 4 and 5 are devoted to the detailed description of both the proposed SQ and the GA-based algorithms for frequency allocation, respectively. The first one is mainly based on the method proposed in (Duque, 1993), though incorporating some improvements concerning the weakest aspects outlined in the conclusions of the work of Duque et al. Important concepts such as the specific *cooling procedure* and the *neighbourhood production* are here described. On the other hand, the genetic algorithm included in section 5 is based on our previous work in the field of NC-based strategies for approaching not only the frequency allocation problem (San José, 2007), but also digital communications (San José, 2008) or image processing problems (San José, 2009), most of them using GAs.
- Finally, the last section of the chapter is devoted to the description of the numerical results, with special emphasis on the comparison between Neural Networks (NNs), SQ

and GAs: advantages and drawbacks of each of them are here explained. For the sake of comparison, a set of well-known problem instances was selected since they have been used in most of the papers related to this problem, thus allowing a direct comparison.

2. The Channel Assignment Problem (CAP)

2.1. Interference constraints

As mentioned in the Introduction, frequency reuse is a key issue in current mobile communication systems. It is well known that the co-channel interference caused by frequency reuse is the most restraining factor on the overall system capacity in wireless networks. Therefore, the main purpose is the simultaneous use of a given radio spectrum while maintaining a tolerable level of interferences. Specifically, each system cell is assigned a set of channels according to the expected traffic demand. This assignment of channels must satisfy the following constraints:

- **Co-site constraint (CSC)**: channels assigned to the same cell must be separated by some minimum spectral distance.
- **Co-channel constraint (CCC)**: the same channel cannot be simultaneously assigned to certain pairs of cells. The *co-channel reuse distance* is the minimum distance at which the same channel can be reused with acceptable interference (Katzela 1996).
- **Adjacent channel constraint (ACC)**: any pair of channels in different cells must have a specified minimum distance (Funabiki, 2000).

The channel assignment algorithm must also take into account the specified traffic profile (number of channels) required in each cell. These non-uniform cell demand requirements imply that those cells with a higher traffic demand will need the assignment of more channels.

2.2. Problem definition

Let us consider the problem where a set of c channels must be assigned to n arbitrary cells (in this work we consider only the *fixed* CAP, where the channels are permanently assigned to each cell; the reader interested in *dynamic* and *hybrid* schemes can see (Gibson, 1996; Hale, 1980; Katzela 1996).

In our problem formulation we assume that the total number of available channels is given –it can be determined by either the available radio spectrum or the lower bound estimated by a graph-theoretic method (Mandal, 2004; Smith, 2000). Without loss of generality, channels can be assumed to be evenly spaced in the radio frequency spectrum. Thus, using an appropriate mapping, channels can be represented by consecutive positive integers. Therefore, the interference constraints are modelled by an *n×n compatibility matrix* **C**, whose diagonal elements c_{ii} represent the co-site constraint, i.e., the number of frequency bands by which channels assigned to cell i must be separated. The non-diagonal elements c_{ij} represent the number of frequency bands by which channels assigned to cells i and j must differ. When this compatibility matrix is binary, the constraints can be expressed more simply: if the same channel cannot be reused by cells i and j, then c_{ij}=1, and, otherwise, c_{ij}=0.

The traffic demand is modelled by means of an n-length demand vector $\mathbf{d}=[d_1,d_2,...,d_n]^T$, whose elements represent the number of channels required in each of the cells. For instance, Fig. 1 shows the four demand vectors that will be used in the simulations' section.

$\mathbf{d}_1^T$ = [1 1 1 3]
$\mathbf{d}_2^T$ = [5 5 5 8 12 25 30 25 30 40 40 45 20 30 25 15 15 30 20 20 25]
$\mathbf{d}_3^T$ = [8 25 8 8 8 15 18 52 77 28 13 15 31 15 16 57 28 8 10 13 6]
$\mathbf{d}_4^T$ = [10 11 9 5 9 4 5 7 4 8 8 9 10 7 7 6 4 5 5 7 6 4 5 7 5]

Figure 1. Traffic demand vectors for the benchmark problems considered.

The assignment to be generated is denoted by an $n \times c$ binary matrix **A**, whose element a_{ij} is 1 if channel j is assigned to cell i, and 0 otherwise. This implies that the total number of 1's in row i of matrix **A** must be d_i (see Fig. 2).

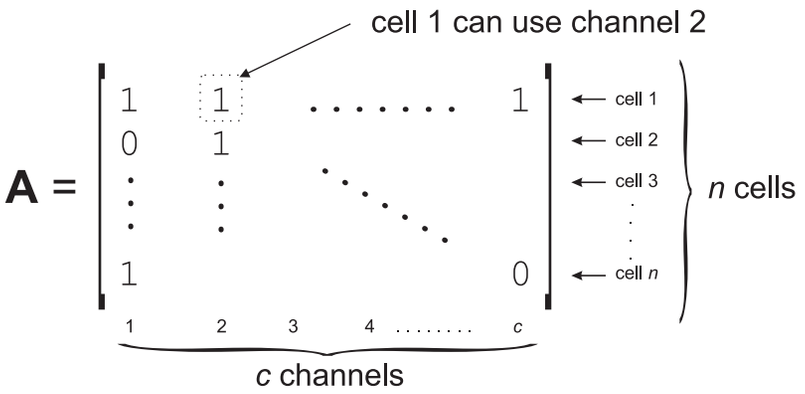

Figure 2. Structure of the allocation matrix **A**.

The cost due to the violation of interference constraints can be written as

$$J_1 = J_{CSC} + J_{ACC} \tag{1}$$

where J_{CSC} and J_{ACC} represent, respectively, the costs due to the violations of the co-site and the adjacent channel constraints. The first one can be written as

$$J_{CSC} = \lambda_{CSC} \sum_{x}^{n} \sum_{i,i\neq j}^{n_x} \sum_{j}^{n_y} \Phi(f_i^x, f_j^x) \tag{2}$$

where parameter λ_{CSC} weighs the relative importance of CSC and $\Phi(f_i^x, f_j^x)$ is a measure of the co-site constraint satisfaction. This parameter equals 0 only if the difference between channels i and j of cell x is $|f_i^x - f_j^x| \geq c_{xx}$, and 1 otherwise. f_α^β represents the assigned frequency for the αth channel of cell β, and n_α is the number of channels in the αth cell.

On the other hand, the cost due to the adjacent channel constraint violation is obtained as

$$J_{ACC} = \lambda_{ACC} \sum_{x\neq y}^{n} \sum_{y}^{n} \sum_{i}^{n_x} \sum_{j}^{n_y} \Psi(f_i^x, f_j^y) \tag{3}$$

where

$$\Psi(f_i^x, f_j^y) = \begin{cases} 0 & \text{if} \quad |f_i^x - f_j^y| \geq c_{xy} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Parameter λ_{ACC} in Eq. (3) is set to weigh the relative importance of the adjacent channel constraint. Finally, the cost due to the violation of the traffic demand requirements is modelled as

$$J_{TRAFF} = \lambda_{TRAFF} \sum_i^n \left(d_i - \sum_j a_{ij} \right)^2 \tag{5}$$

Gathering all the costs, the final cost function to be minimized is

$$J = J_{CSC} + J_{ACC} + J_{TRAFF} \tag{6}$$

If the traffic demand requirements are incorporated implicitly by only considering those assignments that satisfy them, then the cost function can be expressed by $J=J_1=J_{CSC}+J_{ACC}$, subject to $\sum_j a_{ij} = d_i, \forall i$. For that reason, the fitness function to be used in the algorithms is given by $\rho=1/J$.

Finally, the estimation of parameters λ_{CSC} and λ_{ACC} has been achieved using the same inhomogeneous 25-cell network used by Kunz and Lai in (Kunz, 1991) and (Lai, 1996), respectively. After analyzing the number of iterations required for a proper convergence for different values of λ_{CSC} and λ_{ACC}, the optimal values for the weights λ_{CSC} and λ_{ACC} were found to be close to 1 and 1.3, respectively.

It is important to note that the most important difference between different pairs of λ_{CSC} and λ_{ACC} is the required computational load for each of them, since the number of generations required to converge proportionally acts on the execution time. This way, a precise computation of both λ_{CSC} and λ_{ACC} is indispensable to get an efficient allocation algorithm.

3. Simulated quenching algorithm for CAP

3.1. Basic concepts

As mentioned in the Introduction, SQ is methodology proposed to speed up the standard SA algorithms when applied to solve difficult (NP-complete) optimization problems. The original SA method can be viewed as a simulation of the physical annealing process found in nature, e.g., the settling of a solid to its state of minimum energy (ground state). SQ is stronger based on physical intuition though it loses some mathematical rigor.

Generally speaking, an optimization problem consists of a set of S configurations (or solutions) and a cost function J that determines, for each configuration s, its cost $J(s)$. Local

search is then performed by determining the neighbours s' of each solution s. Thus, a neighbour structure $N(s)$ that defines a set of possible transitions that can be proposed by s has to be defined.

When performing local search, in each iteration of the algorithm, a neighbour s' of s is proposed randomly, and s will only be replaced by s' if cost does not increase, i.e., $J(s') \leq J(s)$. Obviously, this procedure terminates in a local minimum that may have a higher cost than the global optimal solution. To avoid this trapping in a suboptimal solution, our proposed SQ method occasionally allows "uphill moves" to solutions of higher cost using the so-called Metropolis criterion (Metropolis, 1953). This criterion states that, if s and $s' \in N(s)$ are the two configurations to choose from, then the algorithm continues with configuration s' with a probability given by $\min\{1, \exp(-(J(s')-J(s))/t)\}$, with t being a positive parameter that gradually decreases to zero during the algorithm. Note that the acceptance probability decreases for increasing values of $J(s')-J(s)$ and for decreasing values of t, and that cost-decreasing transitions are always accepted (see Fig. 3).

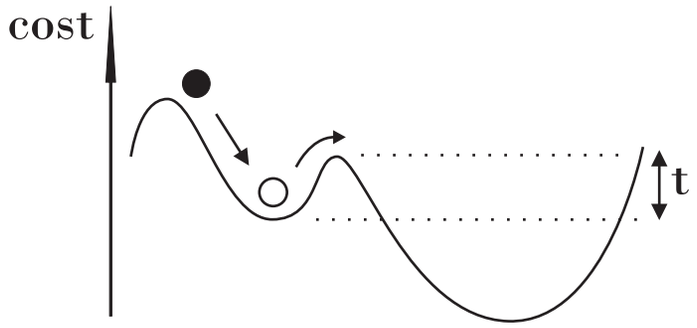

Figure 3. SQ allows uphill moves up to a cost proportional to the instantaneous temperature t.

Mathematically, SA-SQ can be modelled as an inhomogeneous Markov process, consisting of a sequence of homogenous chains at each temperature level t (Duque, 1993). Under this framework, it has been shown (Aarts, 1989; Geman, 1984) that there exist two alternatives for the convergence of the algorithm to the globally minimal configurations. On the one hand (homogenous case), asymptotic convergence to a global minimum is guaranteed if t is lowered to 0, and if the homogenous chains are extended to infinite length to establish the stationary distribution on each level. On the other hand (inhomogeneous case), convergence is guaranteed, irrespective of the length of the homogenous chains, if t approaches 0 logarithmically slow.

The problem arising here is that just the enumeration of the configuration space has an exponential time complexity and, in practice, some approximation is required. The formal procedure is to choose a *cooling schedule* to decide for:

- the start condition (initial temperature, t_0).
- the rule for decreasing the temperature.
- the equilibrium condition.
- the stop condition (final temperature, t_F).

The initial temperature should be chosen high enough in order to allow that most of the proposed transitions pass the Metropolis criterion. Hence, at the start of the algorithm, an explorative search into the configuration space is intended. Later on, the number of accepted transitions decreases as $t \to 0$. Finally, when $t \approx 0$, no more transitions are accepted and the algorithm may stop. As a consequence, the algorithm converges to a final configuration representing the solution of the optimization problem.

As (Duque, 1993) shows, when doing this most cooling schedules lean on the homogenous variant and try to establish and maintain equilibrium on each temperature level by adjusting the length of the Markov chains and the cooling speed.

According to this, the main steps required for solving an optimization problem applying SQ involves that, first, the problem must be expressed as a cost function optimization problem by defining the configuration space *S*, the cost function *J*, the neighbourhood structure *N*. Next, a cooling schedule must be chosen, and, finally, the annealing process is performed.

3.2. Simulated quenching applied to the CAP

In order to apply SQ to solve the CAP, we have to formulate the CAP as a discrete optimization problem, with *S*, *J* and *N* defined. In section 3.2 we have already presented the problem together with its mathematical characterization: a mobile radio network of *n* radio cells, each of them capable to carry any of the *n* available channels. The channel assignment is given by binary matrix **A**, with a_{ij}=1 meaning that channel *j* is assigned to cell *i*. Since the traffic demand is modelled by vector **d,** the total number of 1's in row *i* of matrix **A** must equal d_i.

The cost function *J* is then given by Eq. (6) that quantifies the violation of the interference constraints defined in section 3.1. Thus *J*(*s*) reaches its minimum of zero if all constraints are satisfied.

In this work we will use the same simple strategies for generating the neighbourhood than those used in (Duque, 1993) but with probabilities specifically tuned for our application: (i) *single flip*: just switching on or off channel *i* in cell *j*, –this procedure mimics the mutation operation that will be described later in the GAs context, and (ii) *flip-flop*: replacing at cell *j* one used channel with one unused.

Considering the particularities of the channel allocation problem with hexagonal cells, the same channel should be reused as closed as possible. To approach this goal, the *basic flip-flop* is modified as follows: (ii-1) a cell *j* is chosen at random, (ii-2) from all the channels not used in cell *j*, the channel that is most used within the nearest cells to *j* that may share that channel with cell *j* is switched on, (ii-3) one of the channels previously used at cell *j* is randomly selected and switched off. This *modified flip-flop* is used in conjunction with the basic one.

For the cooling schedule we have implemented of a mixture of different cooling schemes –(Aarts, 1989; Huang, 1986; Romeo, 1989)– with a polynomial-time approximation behaviour. The initial value of the temperature is set to assure a user specified transitions'

acceptance ratio. For that, first, t is set to 0, and then it is iteratively changed until the desired acceptance ratio is reached. Our simulations worked fine with acceptance ratios between 0.55 and 0.6.

Temperature decrement follows a restriction proposed by Huang et al. (Huang, 1986): the decrease ΔJ in the average cost between two subsequent temperatures t and t' should be less than the standard deviation of the cost (on level t). After some calculus (Huang, 1986; Romeo, 1989) this rule is expressed as

$$t' = t \exp\left(-\frac{\lambda t}{\sigma}\right) \tag{7}$$

Since testing for the establishment of equilibrium at a specific t would involve an unacceptable monitoring load, Huang et al. (Huang, 1986) approximate this check in two respects: (i) a Gaussian form for the equilibrium distribution is assumed, whose average and standard deviation are estimated from the Markov chain itself, and (ii) the process is considered stationary if the ratio of the number of accepted transitions, their costs being in a 2δ-legth interval, to the total number of accepted transitions, reaches a stable value $erf(2\delta/\sigma)$. In those cases where the criterion for stationarity can not be reached the length of the chain is bounded proportionally to the number of configurations which can be reached in one transition.

The final temperature is reached if a substantial improvement in cost can no more be expected. In (Huang, 1986; Duque 1993) this is monitored by comparing the difference between the maximum and minimum costs encountered on a temperature level with the maximum single change in cost on that level. If they are the same, the process is assumed to be trapped in a local minimum and the algorithm is stopped.

Numerical experiments show that once being trapped into a suboptimal solution (suboptimal minimum) it is almost impossible to get out of it. Technical literature has described simple approaches to partially improve this situation such as tuning the neighbourhoods to prefer flip-flops which resolve existing interference, or preset violations and to disadvantage those that introduce new ones.

Another solution is based on occasionally allowing arbitrary long jumps while preserving a fast cooling schedule. These long jumps open up the possibility to detrap from any minimum in a single transition, without being questioned by a maybe long chain of acceptance decisions. A simple method for producing these long jumps is to extend the basic transitions –flip-flops– to a chain of consecutive ones. By properly adjusting the chain length, this allows to tunnel through a hill of the cost function landscape in one single jump, instead of painfully working to its top just to fall down into the next valley.

4. Genetic algorithm for CAP

This section describes a low complexity GA (known as μGA) that is applied to solve the channel assignment problem. Next sections present the proposed method, particularizing the concepts to the CAP for a better understanding.

4.1. Basic concepts

The golden rule of GAs is that a set of potential solutions (*population*) can be represented with a predetermined encoding rule. At every iteration k, each potential solution (*chromosome*) is associated to a *fitness* value, $\rho_i(k)$, in accordance to its proximity to the optimal solution. Considering the problem of frequencies assignment, the goal is to avoid violating any of the constraints described in section 3.1, while satisfying the traffic demand in each cell and minimizing the required overall bandwidth.

The initial set of potential solutions, $\mathcal{P}[0]$ (population at k=0), is randomly generated when no *a priori* knowledge of the solution is available (in our specific CAP, some a priori information is available and will help to generate the initial population –see (Lai, 1996). Let us denote $\mathcal{P}[k] = \{\mathbf{u}_i\}_{i=1}^{n_p}$ to the population at iteration k, with n_p being the number of individuals $\boldsymbol{u}_i$ per generation. In our problem, $\boldsymbol{u}_i$ consists of a string structure containing all the channels required for each base station (see Fig. 4). This way, each string represents a particular assignment for all the base stations. By assigning, the number of elements in each string to satisfy the required number of channels for each cell, such computations can be effectively ignored in the objective function.

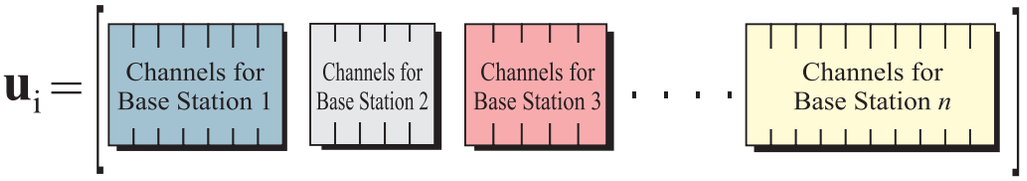

Figure 4. Schematic representation of the chromosome structure for the CAP. Channels for each base station are coded consecutively into the chromosome.

The number of individuals that constitute the population, n_p, is an important issue to be addressed. In (Lai, 1996), Lai and Coghill suggest that a reasonable choice should be in the range 30-110 in order to have a large variability within the population. Since we propose a μGA, the population size would be much smaller, in the range 10-20.

4.2. Genetic operators

Once individuals are generated and given a fitness value, the next step consists in applying the *genetic operators*, mainly *mutation* and *crossover*, to those individuals selected using a fitness-based selection algorithm (Fig. 5 shows the pseudocode that outlines the main steps of the proposed GA). This selection procedure is based on a biased random selection, where the fittest individuals have a higher probability of being selected than their weaker counterparts. This way, the probability of any individual to be selected is $P_i(k) = \rho_i(k) / \sum_{j=1}^{n_p} \rho_j(k)$, with $\rho_i(k)$ being the fitness of the ith individual in the population during the kth iteration of the algorithm. The simplest way to implement this concept is based on a *roulette wheel*, where the size of each slot in the wheel is proportional to the individual's fitness (Mitchell, 1996).

```
0. GENERATE POPULATION P[0]
1. DO{
     1.1 EVALUATE FITNESS OF INDIVIDUALS IN P[k]
     1.2 DO{
            SELECTION (ROULETTE WHEEL)
            APPLY GENETIC OPERATORS:
                  PMX CROSSOVER (PROBABILITY p_c)
                  MUTATION (PROBABILITY p_m)
           }
          WHILE NEW GENERATION NOT COMPLETED
     1.3 EXECUTE CREATE-ELITE
     }
     WHILE {TERMINATION CRITERIA = NO}
2. PRINT BEST ASSIGNMENT SOLUTION u_i

FUNCTION CREATE-ELITE:
{
     SELECTION OF BEST INDIVIDUALS
     MUTATION (PROBABILITY p_m)
     ELITE = MUTATED ELITE + NOT-MUTATED ELITE
     INSERT ELITE INTO POPULATION:
          P[k+1]=INSERT(ELITE,P[k])
}
```

Figure 5. Pseudocode of the proposed GA.

The main genetic operators are:

- **Mutation**: the mutation operator modifies specific individuals with probability p_m, changing the value of some concrete positions in the encoding of individual $\boldsymbol{u}_i$. The number of positions within the encoding of the channels of each cell that are candidates for possible mutation is half the number of channels present at that base station. Both the specific positions and the new values are randomly generated, and mutation is performed with probability p_m. Notice that this genetic operator promotes the exploration of different areas of the solutions space. A low level of mutation serves to prevent any element in the chromosome from remaining fixed to a single value in the entire population, while a high level of mutation will essentially result in a random search. Hence, probability p_m must be chosen carefully in order to avoid excessive mutation. To maintain a good balance between such extremes a good initial value for p_m is 0.02-0.05 (Lai, 1996; Grefenstette, 1986). In our application, $p_m(0)$=0.04.
- **Crossover:** the crossover operator requires two operands (*parents*) to produce two new individuals (*offspring*), which are created merging parents by crossing them at specific internal points. This operation is performed with probability p_c. Since parents are selected from those individuals having a higher fitness, the small variations introduced within these individuals are intended to also generate high fit individuals.

Simulation results show that with the *simple crossover* operator, a significant number of the generated configurations have the same frequency assigned to a group of base stations that interfere with each other. To alleviate this problem we have implemented the *partially matched crossover* operator (Lai, 1996). This operator partitions each string into three randomly chosen portions. When this operator encounters that the same frequency has been

assigned more than once, it solves this conflict by rearranging the conflicting elements in each string –see (Lai, 1996) for a detailed description. Our range for p_c is [0.35, 0.85].

4.3. Elitism, termination criteria and convergence

The proposed algorithm also implements an elitism strategy, where the elite for $\mathcal{P}[k+1]$ is formed by selecting those individuals from both the elite of $\mathcal{P}[k]$ and the mutated elite of $\mathcal{P}[k]$ having the highest fitness value in the population. The mutation of the elite is performed with a probability $p_{m,e}=0.25p_m$. No crossover is performed on the elite.

This elitist model of the GA presents some convergence advantages over the standard GA. Using Markov chain modelling, it has been proved that GAs are guaranteed to asymptotically converge to the global optimum –with any choice of the initial population– if an elitist strategy is used, where at least the best chromosome at each generation is always maintained in the population (Bhandari, 1996).

The whole procedure is iterated until a termination criterion is satisfied. In our simulations, the search is terminated when there are no significant changes between the maximum and minimum values of the objective function in any two successive generations. Notice that, it can not be guaranteed that a valid solution is found in a finite number of iterations. Besides, the time required to compute an optimal solution increases exponentially with the size of the problem (Beckmann, 1999; Kunz, 1991; Funabiki, 1992). Thus, it is necessary to develop approximate methods capable of finding at least a near-optimum solution within a reasonable amount of time.

However, Bhandari et al. provided the proof that no finite stopping time can guarantee the optimal solution, though, in practice, the GA process must terminate after a finite number of iterations with a high probability that the process has achieved the global optimal solution. Note that, in our problem, the optimal string is not necessarily unique and there may be many strings that provide the optimal value (Bhandari, 1996).

In our proposed GA, the coding scheme guarantees that the traffic demand is always satisfied. However, in hard assignment instances it can be impossible to minimize the cost function to zero, i.e., some of the interference constraints may be violated by the generated assignment. In those cases where the optimal solution is not achieved in a finite time, invalid assignments should be resolved by manually assigning more frequencies to the affected cells (thus yielding a span that is larger that the lower limit). Nevertheless, none of the proposed test problem instances led to this situation.

At the end, the string $\boldsymbol{u}_i$ corresponding to the highest fit chromosome is finally chosen as the channel allocation problem solution.

4.4. Diversity and related genetic operators

Convergence properties become notably improved with the introduction of procedures to adjust the parameters in order to achieve and maintain a good population diversity. This

diversity is a crucial issue in the performance of any evolutionary algorithm, including GAs: standard GAs have a tendency to converge prematurely to local optima, mainly due to selection pressure and too high gene flow between population members (Ursem, 2002). A high selection pressure will fill the population with clones of the fittest individuals and it may result in convergence to local minima. On the other hand, high gene flow is often determined by the population structure. In simple GAs, genes spread fast throughout the population, and diversity quickly declines.

On the other hand, one of the main drawbacks of standard GAs is their excessive computational load: the application of the genetic operators is often costly and the fitness function evaluation is also a very time-consuming step. Besides, population sizes n_p normally are 100, 200 or even much higher –for instance, (Ursem, 2002) uses n_p=400 individuals. The method used in this paper works with much smaller population sizes, in the order of 10 to 20 individuals. An elite of 3 individuals is selected and the crossover and mutation probabilities depend on the Shannon entropy of the population (excluding the elite) fitness, which is calculated as

$$\mathcal{H}(\mathcal{P}[k]) = -\sum_{i=1}^{n_p} \rho_i^*(k) \log \rho_i^*(k) \tag{8}$$

with $\rho_i^*(k)$ being the normalized fitness of individual $\boldsymbol{u}_i$, i.e.,

$$\rho_i^*(k) = \frac{\rho_i(k)}{\sum_{i=1}^{n_p} \rho_i(k)} \tag{9}$$

When all the fitness values are very similar, with small dispersion, $\mathcal{H}(\mathcal{P}[k])$ becomes high and p_c is decreased –it is not worthwhile wasting time merging very similar individuals. This way, the exploration character of the GA is boosted, while, conversely, exploitation decreases. On the other hand, when this entropy is small, there exists a high diversity within the population, a fact that can be exploited in order to increase the horizontal sense of search. Following a similar reasoning, the probability of mutation is increased when the entropy is high, so as to augment the diversity of the population and escape from local suboptimal solutions (exploitation decreases, exploration becomes higher). Therefore, we have that probabilities p_m and p_c are directly/inversely proportional to the population fitness entropy, respectively.

Finally, some exponentially dependence on time k is included in the model –making use of exponential functions– in order to relax, along time, the degree of dependence of the genetic operators' probabilities on the dispersion measure.

The complexity of the thus obtained GA is notably decreased since crossover is applied with a very low probability (and only on individuals not belonging to the elite), and the diversity control scheme allows the algorithm to work properly with a much smaller population size.

5. Numerical results

This section evaluates the performance of the proposed algorithms in terms of convergence and solution accuracy under different conditions. Radio base stations are considered to be located at cell centers and the traffic is assumed to be inhomogeneous, with each cell having a different and *a priori* known traffic demand. Following the ideas shown in (Lai, 1996), the initial population is constructed using the available *a priori* information, i.e., the algorithm assigns a valid string of frequencies to all the cells following a simple approach: first, the algorithm attempts to assign a set of valid frequencies to as many base stations as possible. In the event that valid frequencies cannot be located to some of the cells, they are then randomly assigned.

5.1. Benchmark problems

In order to evaluate the performances of the proposed methods and compare them to other approaches, performance is analyzed using the set of thirteen benchmark problems defined in (Sivarajan, 1989) (also used in (Funabiki, 2000) as well as problems 1, 2 and 4 from (Ngo, 1998) (we will refer to these problems with numbers 14, 15 and 16). The characteristics of the first 13 benchmark instances can be found in (Sivarajan, 1989). The definition of problems 14, 15 and 16 are summarized in Table 1, where all the channel demand vectors were shown in Figure 1, and the compatibility matrices are: $\mathbf{C}_1$ (matrix in Example 1, page 846, in (Sivarajan, 1989)), $\mathbf{C}_2$ (matrix in Fig. 3 (c) (Funabiki, 1992), p. 435) and $\mathbf{C}_3$ (matrix in Fig. 3 (a) (Funabiki, 1992), p. 435). The total number of frequencies varies from 11 to 221. Benchmark problem 15 belongs to a particular set of useful benchmark tests for cellular assignment problems called *Philadelphia problems*. Notice that (Sivarajan, 1989) presents some variations from the original Philadelphia problems, which were first presented by Anderson (Anderson, 1973) in the early 70's. These problems constitute, by far, the most common set of benchmark problems for channel assignment algorithms, making it possible to compare the obtained solutions with previously published results. Notice that problems 1–4 and 9–14 consider the three constraints defined in the beginning of section 3.1, while problems 5–8, 15 and 16 consider only the co-channel and co-site constraints.

Problem No.	No. of Cells	Lower Bound	Compatibility Matrix	Demand Vector
14	4	11	$\mathbf{C}_1$	$\mathbf{d}_1$
15	21	221	$\mathbf{C}_2$	$\mathbf{d}_2$
16	25	73	$\mathbf{C}_3$	$\mathbf{d}_4$

Table 1. Specifications of benchmark problems No. 14, 15 and 16.

As an example, Fig. 6 shows the cellular geometry of the Philadelphia problem with n=21 cells (the cluster size for CCC is N_c=7).

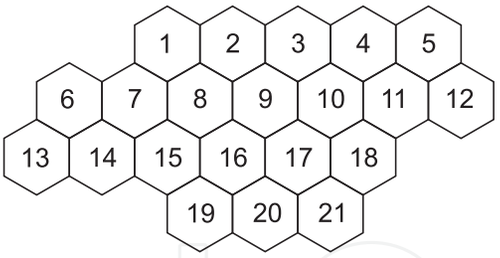

Figure 6. Cellular geometry for the Philadelphia benchmark problem with n=21 cells.

5.2. Adjustment of parameters and convergence performance

In this section, the convergence properties of the proposed methods are studied. Results shown in Table 2 are average values over 25 trials for each problem. The parameters to be set in the GA are: the number of iterations n_g, the initial mutation and crossover probabilities, the population size n_p, and the parameters of functions $p_c(k)$ and $p_m(k)$. After several trials that helped to fine tune the parameters ensuring that the computation is manageable, the optimal values were found to be:

- Number of fitness evaluations: 100 (for problem 14), 25,000 (problems 5–8 and 11), 50,000 (problems 12, 15 and 16), 75,000 (problem 13), 100,000 (problems 1–4), 150,000 (problem 10) and 300,000 (problem 9). If the values corresponding to problems 10, 12, 14 and 16 are compared to those shown in (Ngo, 1998) it can be seen that these values mean a reduction of 75% (in problems 15, 12 and 16) and 62.5% (in problem 10) with respect to the number of iterations needed in (Ngo, 1998); in problem 14 both approaches require 100 iterations. Notice that, since not every offspring needs to be evaluated in each generation, the number of fitness evaluations is a more representative parameter of the performance than the number of generations.
- Initial crossover probability, $p_c(0)$: this parameter is set to 0.35 in problems 5–8 and 11–16, while instances 1–4, 9 and 10 showed better results with 0.25.
- Initial mutation probability, $p_m(0)$: 0.04 for all the problems.
- Population size, n_p: 10 individuals, except in problems 1–4, 9 and 10, which required 20.
- Simulations show that λ_{CSC}=1 and λ_{ACC}=1.3 lead to faster convergence as compared to λ_{ACC}=1. This result is in accordance to (Lai, 1996), where $\lambda_{ACC\text{-}optimal}$=1.1/2 was obtained.

On the other hand, the SQ algorithm has been implemented with a mixture of standard and modified *flip-flops* (described in section 4.2). Problems 5–8, 11–12, 14–16 are solved with a configuration of 50-70% of modified flip-flops, while problem instances 1–4, 9, 10 and 13 used 20-40% of modified flip-flops. The remaining cases, in all problem instances, were implemented with standard flip-flops. To explain this experimental adjustment, just notice that the more complex is the problem instance, the more explorative must be the global search for solutions in order to avoid convergence to suboptimal local minima.

Comparative results are shown in Table 2. The performance is measured using the percentage of convergence to the solutions, defined as the ratio of the total number of

successful convergence to the total number of runs. Table 2 shows the results for problems 10, 12, 14, 15 and 16, whose convergence properties have been previously studied by Ngo and Li using a GA-based scheme (Ngo, 1998) and by Funabiki and Takefuji, who applied a NN-based algorithm to solve these instances (Funabiki, 1992).

	Percentages of convergence (%)			
Problem No.	Funabiki & Takefuji's NN (Funabiki, 1992)	Ngo & Li's MGA (Ngo, 1998)	Proposed μGA	Proposed SQ
10	-	21	24	15
12	23	80	86	78
14	100	100	100	100
15	77	92	90	95
16	9	99	99	98

Table 2. Comparison between convergence results.

Results show that both GA or SQ based procedures outperform the convergence results of the neural network for solving the fixed CAP. The four approaches converge properly in 100% of cases in problem 14. In problems 12, 15 and 16, both genetic methods converge more frequently than the neural network-based approach, and SQ is slightly better than GA in problem 15, while marginally worse in problems 12 and 16. In problem 15 the GA shows a little bit worse convergence results than (Ngo, 1998) (only in about 2%) while SQ moderately improves the MGA. In spite of that, the proposed method involves fewer computational load than that required by (Ngo, 1998) (see Table 3) and the complexity of the SQ method is intermediate between that of the standard GA (MGA) and that of the proposed μGA. In contrast, the μGA presents notably better convergence in problems 10 and 12, where MGA and SQ offer very similar results. In essence, in problems 12, 15 and 16 algorithms exhibit very similar results, with the μGA being less complex.

5.3. Computational complexity

Table 3 shows the execution times required to solve these problems. Bold figures show the CPU time normalized to the time required to solve problem 15 using the μGA.

It can be seen how the computational burden of the proposed method is about 20% lower than that of the standard GA by Ngo and Li (Ngo, 1998) (18% in problem 15, 23% in problem 12, and 20% in problems 10 and 16).

On the other hand, the SQ method shows larger execution times in order to obtain similar convergence figures (as noticed in previous sections). Only in problem No. 15 SQ requires less computational load than the MGA algorithm, although, even in this problem, the μGA obtained the results faster. Notice that this reduction in the computational load observed in

GA-based approaches is achieved maintaining a very similar –or even better– percentage of convergence (Table 2) and with the three approaches getting optimal conflict-free solutions.

	MGA (Ngo, 1998)		μGA		SQ	
10	4.129	**26.12**	3.285	**20.78**	6.101	**38.61**
12	0.959	**6.07**	0.738	**4.67**	1.022	**6.468**
14	0	**0**	0	**0**	0.01	**0.06**
15	0.192	**1.22**	0.158	**1**	0.189	**1.196**
16	0.284	**1.80**	0.226	**1.43**	0.386	**2.443**

Table 3. Execution times (in seconds) for benchmark problems 10, 12, 14, 15 and 16. CPU: AMD Athlon XP 2100+ 1.8 GHz. Bold figures show the CPU time normalized to the time required to solve problem 15 using the μGA

Comparing the values given in Table 3 for (Ngo, 1998) with the specific values reported in the original author's paper, a small difference can be observed. The reason is that the algorithm has been programmed and run in a different computer and language. In order to get the comparative figures shown in Table 3, both methods were similarly programmed and run in the same computer environment.

5.4. Optimal solutions

Now, different search techniques are compared when they run without any time constraint and an optimal solution is guaranteed. Figure 7 shows the execution times for three different algorithms: (i) the IDA (Iterative Deepening A) algorithm (Nilsson, 1998), which offers a quite simple algorithm that can solve large problems with a small computer memory, (ii) the so-called BDFS (Block Depth-Fist Search) real-time heuristic search method proposed in (Mandal, 2004), (iii) the proposed GA, and (iv) the proposed SQ method. For the sake of comparison, we have chosen the same number of cells and number of channels than in (Mandal, 2004).

It can be seen first that the BDFS algorithm produces an increasing average speedup over the IDA method. On the other hand, the proposed μGA outperforms BDFS (and, hence, IDA) whenever the complexity of the problem becomes considerable. In these cases, the running time of the μGA is about 20% smaller than the BDFS. Only in the three simplest cases (a: n=5, c=3), (b: n=5, c=4) and (c: n=7, c=3), the minimum computational load required to implement the μGA is larger than the BDFS, though still much better than the IDA.

When SQ is used, results show that for simple configurations, computational load is approximately that of the GA-based method. However, as complexity (in terms of the number of channels) is increased, the computational load of the SQ procedure tends towards that of the IDA algorithm. These results are in accordance with those outlined in the other numerical simulations.

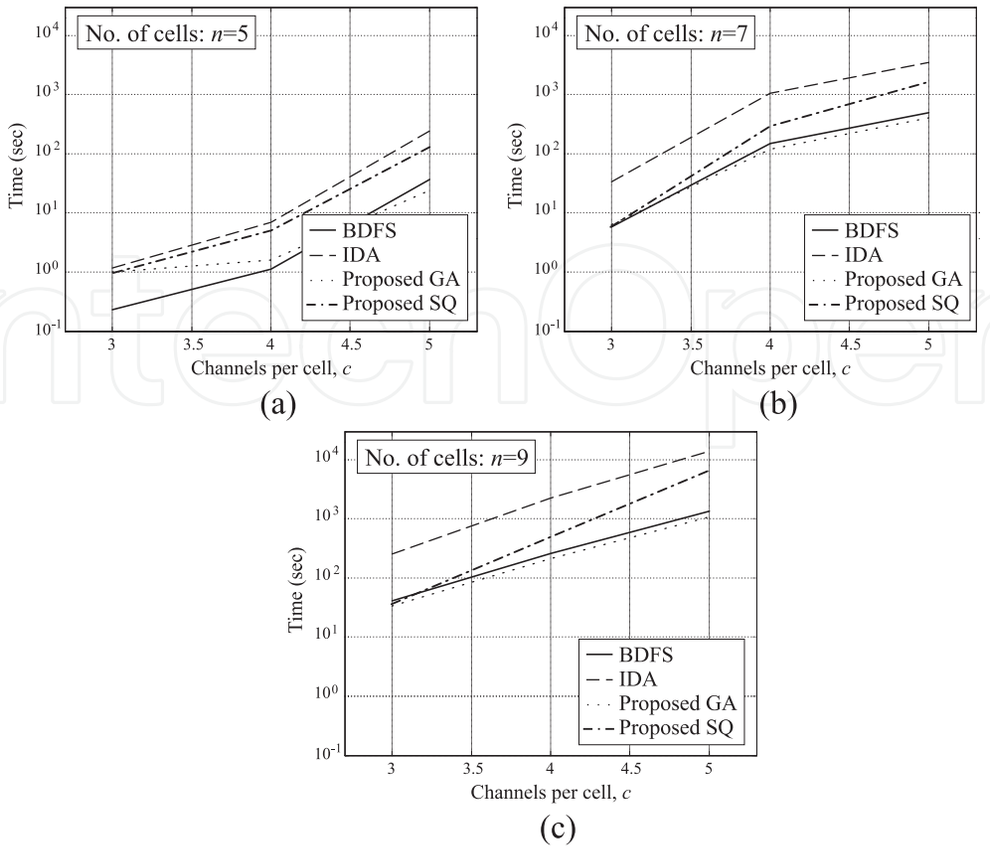

Figure 7. Execution time performance comparison between three different methods for CAP.

5.5. Performance with harder instances

To conclude this analysis of the performance of the proposed methods, two more cases that are more difficult have been studied. Their main characteristics are shown in Table 4.

	No. of Cells, n	**d**	ACC	CCC	CSC
A	21	$2\mathbf{d}_3$	Yes	Yes	Yes (c_{ii}=5)
B	21	$4\mathbf{d}_3$	Yes	Yes	Yes (c_{ii}=5)

Table 4. Definition of benchmark instances A and B.

Our methods are going to be compared with the GA-based approach by Funabiki et al. (Funabiki, 2000). Using the μGA the best assignments we were able to find required 855 and 1713 channels, for problems A and B, respectively, while the method in (Funabiki, 2000) required slightly higher values, 858 and 1724 (results are shown in Table 5). On the other hand, the SQ method requires 855 and 1715 channels, respectively. Thus, in case A –which is a bit simpler than case B- both SQ and GA achieve near optimal results, while in case B –a rather more complex network- SQ requires two more channels than the GA, and 9 less than the NN method. In terms of computation times, the μGA took 11.86 and 23.76 seconds, for problems A and B, respectively; the other GA-based algorithm took about 16.73 and 32.8

seconds, respectively, and, finally, the SQ approach took 15.20 and 37.15 seconds, respectively (see Table 6). This means a reduction in time of 38–41% in favour of the proposed μGA method, while the NN and SQ approaches showed very similar execution times.

	μGA	NN (Funabiki, 2000)	SQ
A	855	858	855
B	1713	1724	1715

Table 5. Best assignments for benchmark instances A and B.

	μGA	NN (Funabiki, 2000)	SQ
A	11.86	16.73	15.20
B	23.76	32.80	37.15

Table 6. Computation time for benchmark instances A and B.

6. Conclusions

NC-based algorithms (GA and SQ) have been proven to fit very well for solving complex NP-complete problems such as the fixed channel allocation problem. Both of them show good convergence properties and reduced computational load. We have solved 18 different benchmark instances with successful results, proving, this way, the accuracy, flexibility and robustness of the proposed methods. Making use of several well-known benchmark instances, their performances have been shown to be superior to those of the existing frequency assignment algorithms in terms of computation time, convergence properties and quality of the solution. Even when compared to one of the best previous approaches –based on a NN-based scheme–, GA and SQ methods have been able to find better solutions to the most complex benchmarks tested.

While both the μGA and SQ offer similar computational load, convergence properties and quality of the solution for simple and moderately-simple benchmark instances, the proposed μGA shows the most reduced computational load when applied to complex problems.

Author details

Luis M. San-José-Revuelta
University of Valladolid, Spain

Acknowledgement

This work has been partially supported by Spanish project TEC2010-21303-C04-04.

7. References

Aarts, E. & Korst, J. (1989). *Simulated annealing and Boltzmann machines*, Wiley (New York).

Anderson, L.G. (1973). A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system, *IEEE Trans. Commun.* Vol. 21 (1973) 1294–1301.

Beckmann, D. & Killat, U. (1999). A new strategy for the application of genetic algorithms to the channel assignment problem, *IEEE Trans. Vehicular Tech.*, 48 (1999) 1261–1269.

Bhandari, D., Murthy, C.A. & Pal, S.K. (1996). Genetic algorithm with elitist model and its convergence, *International Journal on Pattern Recognition and Artificial Intelligence*, Vol. 10, No. 6 (1996) 731–747.

Box, F. (1978). A heuristic technique for assignment frequencies to mobile radio nets, *IEEE Trans. Vehicular Technol.*, Vol. 27 (1978) 57–64.

Černy, V. (1985). Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm, *J. Opt. Theory Appl.*, Vol. 45 (1985) 41-51.

Cuppini, M. (1994). A genetic algorithm for channel assignment problems, *Eur. Trans. Telecommunications and Related Technology,* Vol. 5, No. 2 (1994) 285–294.

Duque-Antón, M. Kunz, D. & Rüber, B. (1993), Channel assignment for cellular radio using simulated annealing, *IEEE Trans. on Vehicular Technology*, Vol. 42 (1993) 14–21.

Funabiki, N. & Takefuji, Y. (1992). A neural network parallel algorithm for channel assignment problems in cellular radio networks, *IEEE Trans. Vehicular Technol.*, Vol. 41 (1992) 430–437.

Funabiki, N., Okutani, N. & Nishikawa, S. (2000). A three-stage heuristic combined neural-network algorithm for channel assignment in cellular mobile systems, *IEEE Trans. Vehicular Technol.,* Vol. 49 (2000) 397–403.

Garey, M.R. & Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York, 1979.

Geman, S. & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *Trans. Patt. Anal. Mach. Intell.*, Vol. 6 (1984) 721—741.

Ghosh, S.C., Sinha, B.P. & Das, N. (2003). Channel assignment using genetic algorithm based on geometric symmetry, *IEEE Trans. on Veh. Techl.*, Vol. 52, No. 4 (2003) 860–875.

Gibson, J.D. (Ed.) (1996). *The Mobile Communications Handbook*, CRC Press.

Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms, *IEEE Trans. Syst. Man and Cybern.*, Vol. 16 (1986) 122–128.

Hale, W.K. (1980). Frequency assignment: theory and applications, *Proceedings of the IEEE* 68 (1980) 1497–1514.

Hopfield, J.J. & Tank, D.W. (1985). Neural computation of decisions in optimization problems, *Biol. Cybern.,* Vol. 52 (1985) 141–152.

Huang, M.D. & Romeo, F., Sangiovanni-Vicentelli (1986). Am efficient general cooling schedule for simulated annealing, Proc. IEEE ICCAD-86, Santa Clara, CA (1986) 381–384.

Ingber, L. (1993). Simulated annealing: practice and theory, *Mathematical and Computational Modeling*, Vol. 18 (1993) 29–57.

Katzela, I. & Naghshineh, M. (1996). Channel assignment schemes for cellular telecommunic-ation systems: a comprehensive survey, *IEEE Personal Commun.*, Vol. 3 (1996) 10–31.

Kim, S. & Kim, S.L. (1994). A two-phase algorithm for frequency assignment in cellular mobile systems, *IEEE Trans. Vehicular Technol.*, Vol. 43 (1994) 542–548.

Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing, *Science,* Vol. 220 (1983) 671–680.

Krishnamachari, B. & Wicker, S.B. (1998). Global search techniques for problems in mobile communications, Wireless Multimedia Laboratory, School of Electrical Engineering, Cornell University, 1998.

Kunz, D. (1991), Channel assignment for cellular radio using neural networks, *IEEE Trans. Vehicular Technol.*, Vol 40 (1991) 188–193.

Lai, W.K. & Coghill, G.G. (1996). Channel assignment through evolutionary optimization, *IEEE Trans. Vehicular Technol.*, Vol. 45 (1996) 91–96.

Lee, A.J. & Lee, C.Y. (2005). A hybrid search algorithm with heuristics for resource allocation problem, *Information Sciences,* Vol. 173 (2005) 155–167.

Lochite, G.D. (1993). Frequency channel assignment using artificial neural networks, *Proc. of the 8th IEE Int. Conf. on Antennas and Propagation*, Vol. 2 (1993) 948–951.

Mandal, S., Saha, D. & Mahanti, A. (2004). A real-time heuristic search technique for fixed channel allocation (FCA) in mobile cellular communications, *Microprocessors and Microsystems*, Vol. 28 (2004) 411–416.

Mathar, R. & Mattfeldt, J. (1993). Channel assignment in cellular radio networks, *IEEE Trans. Vehicular Technol.*, Vol. 42 (1993) 647–656.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953). Equation of state calculations by fast computing machines, *J. Chem. Phys.*, Vol. 21 (1953) 1087-1092.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA.

Ngo, C.Y. & Li, V.O.K. (1998). Fixed channel assignment in cellular radio networks using a modified genetic algorithm, *IEEE Trans. Vehicular Technol.*, Vol. 47 (1998) 163–172.

Nilsson, N.J. (1998). *Artificial Intelligence: a New Synthesis*, Morgan Kaufmann Publisher, Los Altos, CA.

Romeo, F.I. (1989). Simulated annealing: Theory and applications to layout problems, *Memo UCB/ERL M89/29, Univ. California, Berkeley*, (Mar. 1989).

San-José-Revuelta, L.M. (2007). A New Adaptive Genetic Algorithm for Fixed Channel Assignment, *Information Sciences*, Vol. 177, No. 13 (2007) 2655–2678.

San-José-Revuelta, L.M. (2008). A hybrid GA-TS technique with dynamic operators and its application to channel equalization and fiber tracking, in *"Local Search Techniques: Focus on Tabu Search"*, I-Tech Education and Publishing, Vienna, Austria. Editor: W. Jaziri (2008) 106–142.

San-José-Revuelta, L.M. (2009). Fuzzy-aided tractography performance estimation applied to Brain magnetic resonance imaging, *Proc. 17th European Signal Processing Conference (EUSIPCO 2009)* (2009) Glasgow, Scotland.

Sivarajan, K.N., McEliece, R.J. & Ketchun, J.W. (1989). Channel assignment in cellular radio, *Proc. of the 39th IEEE Vehicular Technol. Conf.* (1989), 846–850.

Smith, D.H., Hurley, S. & Allen, S.M. (2000). A new lower bound for the channel assignment problem, *IEEE Trans. on Veh. Tech*., Vol. 49 (2000) 1265–1272.

Ursem, R.K. (2002). Diversity-guided Evolutionary Algorithms, *Proc. Int. Conf. on Parallel Problem Solving from Nature VII (PPSN VII),* (2002) Granada, Spain, 462–471.

IntechOpen